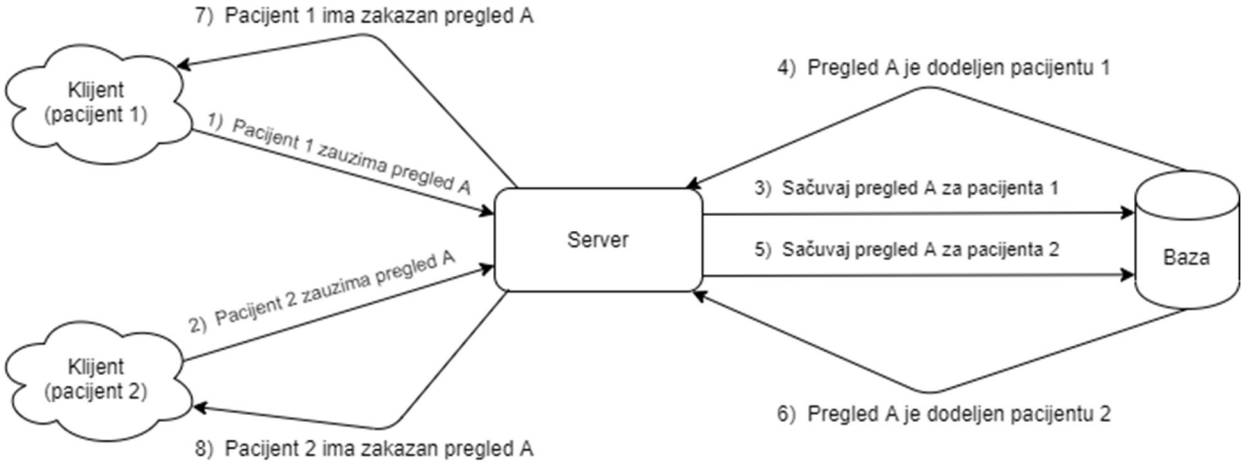
Konfliktne situacije – Student 1

Ime i prezime: Nemanja Jevtić
Broj indeksa: SW 45/2017

1. Konkurentno zakazivanje pregleda u istom terminu kod istog lekara od strane više pacijenata

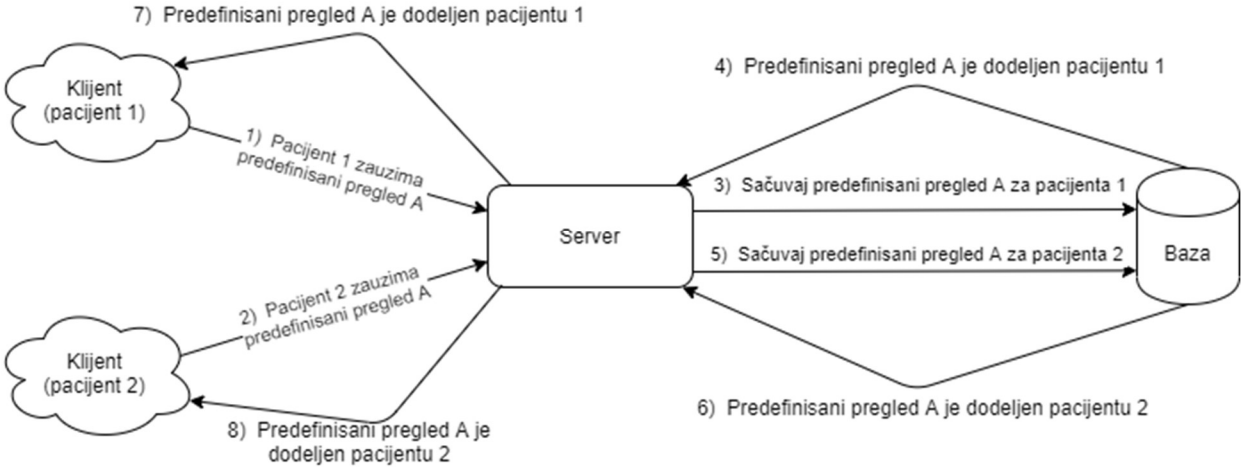
Opis problema: Pacijenti imaju mogućnost da odaberu željenu kliniku pretragom. Odabirom klinike, prikazuju im se svi doktori koji prolaze prethodno izvršenu pretragu. Ta lista doktora se može dalje pretraživati. Na kraju pacijent ima mogućnost odabira željenog lekara i time mu se prikazuje lista slobodnih termina. Klikom na odabrani termin pacijent šalje zahtev za zakazivanje tog pregleda na server. Problem nastaje kada dva ili više pacijenata pošalje zahtev za zakazivanje pregleda kod istog lekara u jednom te istom terminu. Tako se kreiraju dva ili više zahteva za pregledom za koje administrator klinike (ili sam server) treba da dodeli salu. I tako se dobija nekonzistentno stanje, jer je jedan te isti lekar prisutan na više različitih pregleda istovremeno.



Opis rešenja: Konfliktna situacija je rešena primenom optimističnog zaključavanja. Svaki put kada pacijent pošalje zahtev za pregled, na serveru se kreira novi objekat klase *AppointmentRequest* u koji se između ostalog smešta referenca na postojećeg doktora. Anotiranje klase *AppointmentRequest* sa *@Version* neće pomoći jer se svaki put kreira novi objekat. Zato je neophodno anotirati klasu *Doctor* sa *@Version*. Međutim, *hibernate* ne dozvoljava da se podkasa anotira sa *@Version* i zato je neophodno tako anotirati korensku klasu *User*. Kako se kreiranjem novog objekta klase *AppointmentRequest*, i dodavanjem reference na postojećeg doktora, sam objekat doktora ne menja, onda se ne menja ni atribut anotiran sa *@Version*. Iz tog razloga je neophodno dodati atribut *counter* u klasi *User*. Na taj način kada se dodaje postojeći doktor u novi objekat klase *AppointmentRequest*, attribut *counter* u doktoru se poveća za jedan i tako se promeni vrednost atributa anotiranog sa *@Version*. Svaka naredna transakcija koja pokuša da uradi *commit* u bazi, dobiće izuzetak, jer vrednosti atributa anotiranog sa *@Version* kod doktora u bazi i u tim transakcijama neće biti ista. Tako je rešen problem kreiranja više zahteva za pregled kod istog doktora u istom terminu.

2. Konkurentno zakazivanje predefinisanog pregleda od strane više pacijenata

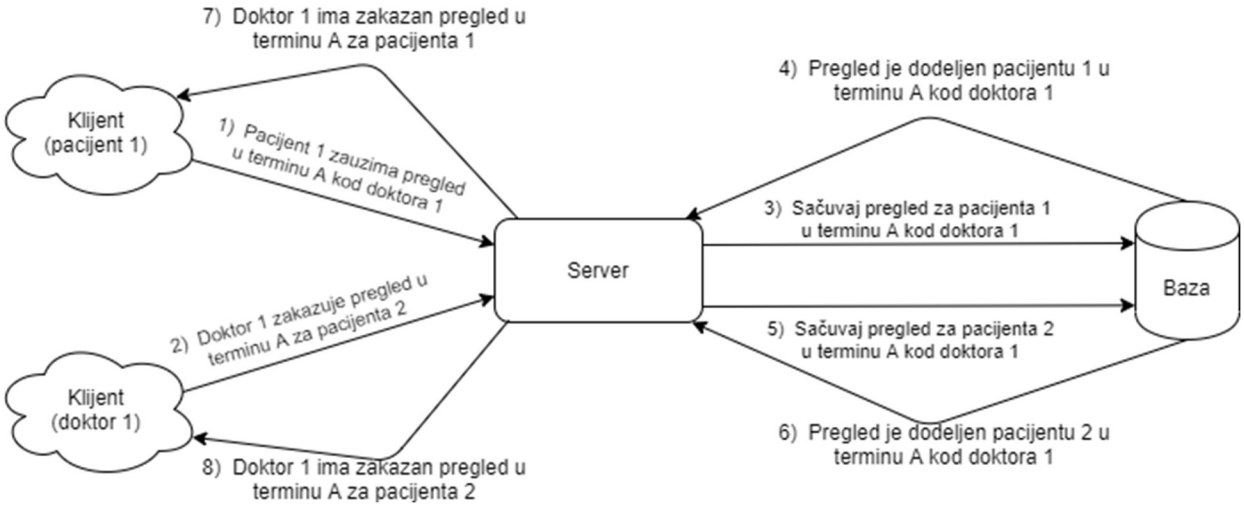
Opis problema: Pored zakazivanja pregleda putem pretrage klinika i doktora, pacijent ima mogućnost zakazivanja pregleda koji su unapred definisali administratori klinike. Problem nastaje kada više pacijenata istovremeno zakažu jedan te isti predefinisani pregled. Tako se dobija nekonzistentno stanje, jer će se na serveru stvoriti dva ili više zahteva za pregled za isti termin kod istog doktora, ali za različite pacijente.



Opis rešenja: Kao i u prethodnom rešenju, i ovde se kreira novi objekat klase *AppointmentRequest*, samo što sada njegova referenca na pregled neće biti *null* kao u prethodnom rešenju. Ali zato ovde se ne možemo osloniti na atribut anotiran sa *@Version* u klasi *Doctor* iz razloga što je moguće da dva pacijenta zakažu dva predefinisana pregleda kod istog doktora, ali u različitim terminima. Ako bi se na njega oslonili, onda bi promenom vrednosti atributa označenog sa *@Version* u jednoj transakciji, bio onemogućen komit druge transakcije iako možda situacija nije konfliktna, jer se radi o istom doktoru, ali različitom terminu. Zato ćemo klasu *Appointment* proširiti sa atributom anotiranim sa *@Version* i atributom *counter*. U jednoj transakciji kada se zakazuje predefinisani pregled, taj *counter* povećati će se povećati, samim tim i atribut anotiran sa *@Version*, i onda sačuvati u bazi. Svaka naredna transakcija koja pokuša da uradi *commit* u bazi, dobiće izuzetak, jer vrednosti atributa anotiranog sa *@Version* kod objekta klase *Appointment* u bazi i u tim transakcijama neće biti ista. Na taj način sprečavamo zakazivanje jednog te istog predefinisanog pregleda za dva različita pacijenta.

3. Konkurentno zakazivanje pregleda u istom terminu od strane lekara i pacijenata

Opis problema: Doktor u toku pregleda ima mogućnost da zakaže naredni pregled u nekom terminu za pacijenta kojeg trenutno pregleda. Istovremeno, drugi pacijent ima mogućnost da zatraži pregled kod u istom terminu kod istog doktora. Tako opet dolazimo u situaciju da jedan lekar ima zakazana dva pregleda u jednom te istom terminu.



Opis rešenja: I ovaj problem je rešen primenom optimističnog zaključavanja. Kada doktor B zakaže novi pregled u terminu A za trenutno pregledanog pacijenta, kreira se objekat klase *AppointmentRequest* kome se dodeljuje referenca na doktora B koji vrši pregled. Objekat klase *AppointmentRequest* se kreira i kada pacijent hoće da zakaže pregled u istom terminu A kod istog doktora B. Dakle, neophodno je proširiti klasu *Doctor* sa atributom *@Version* i atributom *counter* i za svako dodeljivanje reference doktora u objekat klase *AppointmentRequest* treba povećati atribut *counter* i na taj način se menja atribut *@Version.* Ovo rešenje je već primenjeno prilikom rešavanja konflikta broj 1.